10'00"00 インターネット10分講座

MIN SEC 1/100S INTERNET

SSL/TLS 20年の歩みと動向

昨年2014年は、SSL（Secure Sockets Layer）とTLS（Transport Layer Security）というプロトコルがリリースされてから20年が経過し、HeartBleedやPOODLEなどの脆弱性でも話題となった年でもありました。今回の10分講座では、SSL/TLS暗号通信プロトコルの動向を紹介します。

◆ SSL/TLSとは

SSL/TLSは最も普及している暗号通信プロトコルの一つで、TCP/IPの4レイヤーモデルのトランスポート層とアプリケーション層との間に位置するため、広く使われているHTTPばかりでなく、SMTPなど任意のプロトコルを安全に送受信する目的で使用することができます。特にWebにおいて暗号通信機能を提供できるようになったことにより、オンラインショッピング、オンラインバンキングやユーザー認証を必要とする各種オンラインサービスの普及に重要な役割を担ってきました。

SSL/TLSには暗号化、認証、改ざん検知の三つの主要な機能があります。暗号化により通信の盗聴を防ぐことができ、認証によりサーバやクライアントが正しい通信相手であるか確認することができ、改ざん検知により通信中のデータの不正あるいは障害によるデータの誤りを検知することができます。

◆ SSL/TLSの歩み

SSL 1.0プロトコルは、1994年にネットスケープコミュニケーションズ社により開発されましたが、公開前に設計に問題が発見されたことで公開はされず、同1994年にSSL 2.0として公開され、翌1995年にはセキュリティ上の問題を修正したSSL 3.0が公開されました。ブラウザとしてはNetscape Navigator 1.1や、Microsoft® Internet Explorer 2.0に初めてHTTPS機能が搭載され、サーバ側でもApache HTTP ServerをHTTPS対応にするための、OpenSSLの前身となったライブラリ、SSLeayがリリースされ、SSLの普及が大いに加速されました。

1996年以降、SSLの標準化はネットスケープコミュニケーションズ社からIETF TLSワーキンググループに移管され、1999年にSSL 3.0とほぼ同等のRFC 2246 TLS 1.0が、2006年にはCBCブロック暗号モードに対する攻撃への対応などセキュリティ機能を強化したRFC 4346 TLS 1.1が、2008年には暗号アルゴリズムの移行のためにSHA-256、GCMなどを導入したRFC 5246 TLS 1.2がリリースされました。2015年1月現在では、TLS 1.3の策定が行われています。

SSL/TLSのプロトコル、実装ライブラリ、対応Webブラウザ、各種脆弱性や問題の、20年にわたる歴史を年表に整理しましたのでご覧ください。

◆ SSL/TLSの課題と議論

2014年は、ShellShockなどさまざまな脆弱性が発見された年でしたが、その中で、SSL/TLSについてもHeartBleedやPOODLEなどの脆弱性があった年でした。前述の年表をご覧いただくとお気づきになるかもしれませんが、2005年までは単にソフトウェアのアップデートやパッチを適用し、弱い暗号からの移行に対応していればセキュリティ対策として十分であったものが、今回は事情が異なり、より複雑になってきているように思います。これらの問題は、大きく四つに分類することができます。

(1) ソフトウェアの実装上の問題やバグ
(2) 暗号アルゴリズムの危殆化（きたいか）※への対応
(3) 認証局の運用上の問題
(4) SSL/TLSプロトコルの設計上の問題

※暗号アルゴリズムの安全性のレベルが低下した状況、または、その影響により、暗号アルゴリズムが組み込まれているシステムなどの安全性が脅かされる状況を指します

(1) のソフトウェアの実装上の問題については、使用するソフトウェアやライブラリの脆弱性情報を注視し、適切にソフトウェアアップデートやパッチを適用することで解決できます。

(2) の暗号アルゴリズムについては、米国の暗号輸出規制に対応した暗号はもちろんのこと、MD5、DES、RC4などの今となっては弱い暗号の利用を控える必要があります。BEASTなどCBCブロック暗号モードに関連した脆弱性が多く発見され、GCMモードなどの認証暗号を使った暗号が推奨されています。ただし、フィーチャーフォンや古いゲーム機など後方互換性のために、3DESやRC4などの古い暗号スイートを使わざるを得ないケースもあるので注意が必要です。さらに、SHA-1を使用したSSLサーバ証明書についても、Google Chrome™やFirefoxなどのブラウザ、Microsoft製品では2017年1月以降使用できなくなるため、サーバ管理者の方は、SHA-2アルゴリズムを使ったサーバ証明書の移行計画を立てておく必要があるでしょう。

(3) については、2011年頃はDigiNotar社やTURKTRUST社など、SSLサーバ証明書を発行する認証局が攻撃を受け不正な証明書を発行してしまう事件が多発し、認証局の信頼が損なわれました。それまでは証明書失効リスト（CRL）やOCSP（Online Certificate Status Protocol）などPKIの失効の仕組みだけで証明書が信頼できるか判断できたのが、それだけでは十分ではないとされたのです。不正に発行された証明書を検知するための仕組みとして、インターネットドラフト“Public Key Pinning Extension for HTTP”や、“RFC 6962 Certificate Transparency”など、SSL/TLSを補強する技術が実装されつつあります。

(4) についても2010年以降、BEAST、CRIME、POODLEなどSSL/TLSのプロトコル設計上の欠陥をつく攻撃が顕著になってきています。このような場合では、プロトコルを正しく実装したソフトウェア自体には問題はないので、ソフトウェアアップデートでは対応できません。多くの場合、暗号スイートの設定、圧縮設定の無効化、SSL/TLSプロトコルバージョンの設定など、サーバやクライアントの設定により対策する必要があります。

◆ 脆弱性の歴史

さて、ここからはSSL/TLSに関連した主要な脆弱性の歴史について、振り返ってみたいと思います。

1）MD5不正CA証明書（2008）

2007年にMD5のハッシュ衝突攻撃が発表されましたが、これを応用して、MD5 with RSA署名アルゴリズムのSSLサーバ証明書を発行する認証局を利用して、不正なCA証明書に作り変えてしまうという攻撃方法が、Alexander Sotirov氏らの研究グループによって発表されました。発行される証明書のシリアル番号が予測できれば、検証で無視されるフィールドをうまく作り込むことにより、不正な中間CA証明書が作れてしまうというものでした。中間CA証明書を不正作成できたということは、そこから任意のドメイン名のSSLサーバ証明書を発行できるということです。不正CA証明書のハッシュ衝突を計算するために、SONY Play Station 3 200台で構成されたクラスタで、わずか1、2日で計算できたということでも話題となりました。

2）DebianのOpenSSL RSA鍵生成問題（2008）

2008年に、Linuxのディストリビューションの一つであるDebianが提供する、OpenSSLのパッケージのみに発生した脆弱性がありました。直接的なSSL/TLSの脆弱性ではありませんが、SSLサーバ証明書の再発行を必要としたサイトもあり、相当数の影響がありました。DebianのOpenSSLでは、疑似乱数の初期値が固定になっており、生成されるRSA鍵の種類が非常に限定的になっていたことがわかりました。そのため、すべての場合を調べれば、生成される可能性のある秘密鍵のすべてが得られるようになっていました。このような方法で他人が秘密鍵を持っている恐れのある公開鍵はブラックリスト化され、問題がある鍵かどうかのテストサイトもいくつか公開されました。

3）証明書識別名のNULL終端問題（2009）

C言語ではNULL文字（“\0”）を文字列の終端記号として扱いますが、SSLサーバ証明書のドメイン名を表す証明書の識別名では、NULL文字を途中で使うことができます。例えば、攻撃者がwww.example.com用のフィッシングサイトを作りたいとしても、example.comドメインの管理権限はないので通常なら認証局は証明書を発行しませんが、攻撃者がevilsite.jpドメインのオーナーであるとすれば、以下のような識別名の証明書は発行してもらえる可能性があります。

CN=www.example.com\0.evilsite.jp, O=Evil Site, C=US

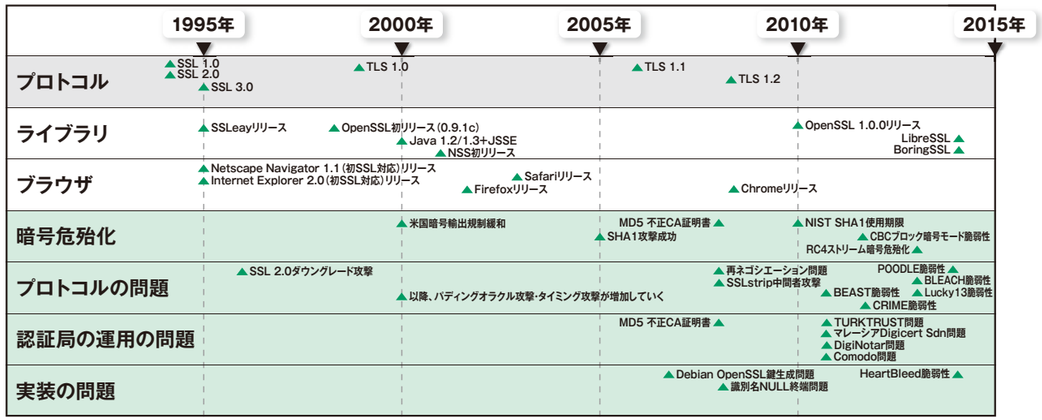
この問題が発生した当時の一部のブラウザでは、NULL文字を文字列の終端記号であるとして扱ってしまったため、上のような名前の証明書をwww.example.comドメイン用であると勘違いしてSSL通信を開始してしまいました。この脆弱性はBlackHat 2009でDan Kaminsky氏らのグループにより発表され、後にすべてのブラウザのアップデートで問題解決されました。

4）再ネゴシエーション問題（2009）

2009年にSSL 3.0以降で導入された再ネゴシエーションのプロトコルに設計上の脆弱性があり、クライアント側の要求に対して任意のデータが挿入できる脆弱性が発見されました。この攻撃に対する緩和策は、サーバが再ネゴシエーションを禁止するか、“RFC 5746 Transport Layer Security（TLS）Renegotiation Indication Extension”というTLS拡張を使用することです。問題発覚以降のアップデートでは、すべてのサーバ、ブラウザ、ライブラリでこのセキュアな再ネゴシエーション機能が有効になっています。

5）DigiNotar不正証明書発行事件（2011）

2011年にオランダ政府の証明書発行業務も行っている、オランダの認証局DigiNotar社がハッキング攻撃を受け、不正に531枚のSSLサーバ証明書を発行するという事件がありました。2011年か



図：SSL/TLS20年の歴史

ら2012年にかけては、英国Comodo社、マレーシアDigicert Sdn社、トルコTURKTRUST社など、認証局のシステムの不備をついたハッキング攻撃により、不正な証明書を発行してしまうという事件が多発しました。その中でも、DigiNotar社の事件は最大級のものでした。

この事件は、運用監査なども適切に実施し、運用上不正な証明書を発行するなどあり得ないと考えられていた認証局が、Google、FacebookやMicrosoftなどの著名サイト用のワイルドカードドメインの証明書を不正発行してしまい、中間者攻撃により通信を盗聴された可能性が高いというもので、認証局の信頼を完全に損なわせるような事件でした。

6）BEAST（2011）

BEASTは、2011年にThai DuongとJuliano Rizzoらの研究グループにより発見された脆弱性で、CBCブロック暗号モードであれば、AESなどの強い暗号でもセッションクッキーが解読され、ハイジャックされる可能性があるというものです。実際にJavaアプレットの脆弱性と組み合わせて、paypal.comサイトに対してハイジャックのデモを行いました。一時、BEAST対策としてRC4ストリーム暗号を使うことが推奨されましたが、RC4暗号にも脆弱性が発見されたため、現在ではCBCモードの初期ベクタ（IV）の処理が変更されたTLS 1.1を使うか、GCMモードなどの認証暗号を使うしか解決策がありません。

7）CRIME（2012）

CRIMEは2012年に発見された、TLSのデータ圧縮機能を使ってセッションクッキーを解読する攻撃です。同じ文字であれば、圧縮後のデータサイズが小さくなる圧縮の性質を利用して、さまざまなデータを埋め込むことによりデータサイズの変化を見ながら、クッキーのすべてを解読しセッションハイジャックする攻撃です。この攻撃は、AESやGCMやSHA-256など強固な暗号を利用していたとしても、使用する暗号に関係なく攻撃が可能であるという特徴があります。サーバの設定によりTLSデータ圧縮を無効化することにより攻撃の緩和が可能です。現在策定中のTLS 1.3では、TLSデータ圧縮は無効化されています。

8）HeartBleed（2014）

2014年に発見された当時のOpenSSL 1.0.1系のバージョンすべてに影響する脆弱性で、認証クッキーやパスワードなどメモリ上の情報が、外部から取得可能となる脆弱性です。これは、“RFC 6520 Transport Layer Security（TLS）and Datagram Transport Layer Security（DTLS）Heartbeat Extension”というTLS拡張の、OpenSSLによる実装の不備をついたものです。Heartbeat（死活確認）で要求されるデータサイズは本来小さいものですが、クライアント側で大きな値を設定すると、サーバ側でメモリの境界を越えて結果を返してしまい、場合によってはクライアントにセッションクッキーやユーザーのパスワードなどの情報が見えてしまう可能性があるというものでした。OpenSSLのアップデートにより、問題は解決しています。ただ、OpenSSLの古いメモリ管理により起きた問題で、今後も同様の脆弱性が発見される可能性があるため、OpenSSLを作り直そうという動きが生まれ、LibreSSLやBoringSSLなどの、OpenSSLのリプレースをめざすプロジェクトが発足しました。

9）POODLE（2014）

POODLEは、2014年に発見されたSSL 3.0プロトコルの、パディングに関する設計上の不具合をついた攻撃で、CBCブロック暗号モードを使っている場合に、セッションクッキーを盗聴される恐れがあります。この脆弱性はTLS 1.0以降では発生せず、SSL 3.0でのみ発生するものです。SSL 3.0とTLS 1.0では、プロトコルにあまり違いがないと言われていますが、小さな違いの一つとしてパディング生成があります。ブロック暗号では決められたブロックサイズの倍数でデータを送りますが、ブロックサイズの倍数にならない場合には、何らかのデータで「詰め物」をし、倍数になるように調整します。これをパディングと呼びます。TLS 1.0では決められたパディング値を使いますが、SSL 3.0では任意の値をパディングに使えます。これを利用して、さまざまな値を埋め込んでみてクッキーの値を推測するのです。POODLE攻撃を緩和するには、TLS 1.0以上を使う、もしくはブロック暗号GCMモードやストリーム暗号を使うなどの方法があります。TLS 1.0では、影響が全くないとされていましたが、TLS 1.0であってもパディングの正しさを確認せず、POODLE攻撃の影響を受ける実装が発見されました。

◆ SSL/TLSの注目すべきトピック

1）LibreSSLとBoringSSL

OpenSSLのHeartBleed脆弱性をきっかけに、これまでOpenSSLで問題視されていたソースコードの複雑さの解消と、今後脆弱性が含まれにくくするために、ソースコードの改変をしたいという動きが生まれました。その一つが、OpenBSDが開発に着手したLibreSSLです。

LibreSSLには次に示すような、大きな特徴があります。

・OpenSSL 1.0.1gからソースコードを分岐させた
・OpenSSLと公開APIが完全互換
・メモリ管理を独自のものではなくシステムコールに置き換え、メモリ関連の脆弱性を低減
・現時点では暗号スイートがOpenSSLより古いが、今後、新しい強固な暗号スイートをサポートする
・古いプラットフォームのサポートを廃止

また、同様にGoogle社も、OpenSSLからソースコードを分岐させた、独自のBoringSSLの開発を開始しました。将来的には、Google ChromeやAndroid™などでOpenSSLの使用を止め、BoringSSLへ移行するよう計画しています。

2）PFS対応

エドワード・スノーデン氏の暴露により、米国国家安全保障局（NSA）が米国全国民の通信を盗聴していたという疑惑が語られており、SSL/TLS暗号通信であっても、暗号化されたままの通信データを巨大なデータベースに保管しておけば、将来、サーバの廃棄されたハードディスクから盗んだり、秘密鍵を解読したりすることにより、SSL/TLS暗号通信をすべて解読できる可能性があります。

将来、サーバ証明書の秘密鍵が漏洩したとしても、通信内容を解読されることはないようにすることを、Perfect Forward Secrecy（PFS）と呼んでいます。具体的には、ECDHEやDHEを含む暗号スイートを使用することで、PFSに対応することができます。ECDHやDHなどの鍵交換アルゴリズムと比較して「E、ephemeral（つかの間の、短命な）」がついています。TLS 1.3では、ECDHEやDHE などPFS対応の暗号スイートを使用します。

DHEやDHで使用されるDiffie-Hellman鍵共有アルゴリズムの実装では、現時点で弱いとされている1,024bit以下のDH鍵を使用する実装が多いためにDHEを使うのは問題があり、ECDHEを使う方がよいと考えられます。

3）SHA-2サーバ証明書への移行

現状では、数多くのサイトでSHA-1 with RSA署名アルゴリズムのサーバ証明書を使っていますが、米国の標準化機関である米国立標準技術研究所（NIST）の暗号アルゴリズムの移行に関するガイドライン SP800-131Aでは、SHA-1を2013年12月31日以降署名と検証には使ってはならないとしています。しかしながら、米国政府のサイトですら、多くがSHA-1の証明書を使用しています。

このような状況を受けて、Microsoft社はすべての製品において2017年1月1日以降、SHA-1証明書を使用した際にエラーとするというポリシーを公表しました。また、Google Chromeでは、2014年11月より段階的にSHA-1証明書を使用した場合にアラートを表示するようにし、Microsoft社製品と同様に2017年1月1日以降にエラー表示になります。Firefoxも同様のポリシーを公表しています。

すべてのサーバ管理者の方は、2016年末までにSHA-256 with RSAなどのSHA-2証明書へ移行する必要があります。証明書の有効期限が1年や2年であるケースが多いことを考慮すると、2015年中にはSHA-2証明書への移行を検討しなければなりません。レガシーな環境では、SHA-2証明書へ対応できないケースもあるでしょうから、そのような場合には、システム全体の更改も検討しなければなりません。

4）Certificate Transparency

Certificate Transparencyは、DigiNotar社やTURKTRUST社などの認証局への攻撃による不正証明書発行事件を受けて、2013年にExperimental RFC 6962として公開された、不正に発行された証明書を検証するための仕組みの一つです。認証局が、不正な意図しない証明書を発行していないことを示す監査記録を公開サーバに公表することにより、あるサイトに不正な証明書が別途出ていないかを判定することができます。

既に、Google ChromeがCertificate Transparencyに対応しており、例えばCertificate Transparencyに対応した、DigiCert社のサイト（https://www.digicert.com）を表示した場合に、「公開監査が可能です。」と表示されます。一方、対応していないサイトでは「公開監査記録がありません。」と表示されます。

5）Public Key Pinning

Public Key Pinning、もしくは Certificate Pinningもまた、DigiNotar社のような不正証明書発行事件から利用者を守るための技術で、“Public Key Pinning Extension for HTTP”というスタンダードトラックのインターネットドラフトになっており、既に、Google ChromeやFirefoxで実装されています。

サイトの証明書のハッシュ値を、何らかの方法で取得し、それとサイト証明書とを比較し、一致していなければ警告を出すというものです。サイト証明書のハッシュを提供する方式としては「ブラウザに組み込む」と「サイトのHTTPヘッダでハッシュ値を提供する」という、二つの方式があります。

GoogleやFacebookなど著名なサイトを閲覧する場合には、ブラウザ組み込みのPinningのハッシュ値を使うことができます。そうでないサイトの場合には、HTTPヘッダ方式を使用します。

6）OCSP Stapling

“RFC 6960 X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP”は、あるサーバ証明書が失効しているかどうかを確認するためのプロトコルで、現在多くのブラウザでは、もう一つの失効検証方法である、CRLよりも優先して使用される失効検証方法です。ただ、OCSPには現在二つの問題があると言われています。

・OCSPレスポンスが取得できない場合でも、多くのブラウザが検証成功としてしまう。DoS攻撃により検証妨害した場合に偽サイトでも有効としてしまう可能性がある
・サイトのアクセス記録がOCSPレスポンダサーバ、つまり認証局側にも残ってしまい、プライバシー上の問題がある。アクセス記録は本来Webサイトしか持つべきでない

これらの問題を解決するのが“RFC 6066 Transport Layer Security（TLS）Extensions: Extension Definitions”で規定されたOCSP Staplingです。Staplingとはホッチキス留めのことで、TLSのセッションにおいて拡張領域にOCSPによる、証明書の状態を入れることができます。OCSP Staplingを使用することにより、前述のOCSPの問題を解決することができます。

・サイトに接続できれば必ず証明書の失効状態を確認でき、認証局のOCSPに接続する必要がない
・失効検証のために認証局のOCSPレスポンダに接続する必要がないので、認証局にはアクセスログは残らず、プライバシーの懸念がない

OCSP Staplingは多くのブラウザ、サーバで対応しており、サーバに設定を行うだけで利用可能になります。

◆ おわりに

SSLの誕生から20年が経ち、SSL/TLSが社会インフラの重要なプロトコルとなったと同時に、さまざまなセキュリティ上の問題も顕在化してきています。今後も、こうした動向に注目しながら安全にSSL/TLSを利用していく必要があるでしょう。

（富士ゼロックス株式会社 漆嶌 賢二）